

# **The Final Report**

**Title: Construct Abstraction for Automatic Information  
Abstraction from Digital Images**

**Principal Investigator:**

Masanori Sugisaka, Ph.D., Professor  
Department of Electrical and Electronic Engineering  
Oita University  
700 Oaza Dannoharu 870-1192  
Oita, Japan  
Telephone: +81-097-554-7831  
Facsimile: +81-097-554-7818  
E-mail: msugi@cc.oita-u.ac.jp

**Contract Number:** FA5209-05-P-0171

**AOARD Reference Number:** AOARD-54063

**AOARD Program Manager:** Tae-Woo Park, Ph.D.

**Period of Performance:** 15 December 2004 – 1 May 2006

**Submission Date:** 30 May 2006

Report Documentation Page			Form Approved OMB No. 0704-0188		
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE <b>13 JUN 2006</b>		2. REPORT TYPE <b>Final Report (Technical)</b>		3. DATES COVERED <b>15-12-2004 to 01-05-2006</b>	
4. TITLE AND SUBTITLE <b>Construct Abstraction for Automatic Information Abstraction from Digital Images</b>			5a. CONTRACT NUMBER <b>FA520905P0171</b>		
			5b. GRANT NUMBER		
			5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) <b>Masanori Sugisaka</b>			5d. PROJECT NUMBER		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>Oita University,700 Oaza Dannoharu,Oita 870-1192,Japan,JP,870-1192</b>			8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) <b>The US Resarch Labolatory, AOARD/AFOSR, Unit 45002, APO, AP, 96337-5002</b>			10. SPONSOR/MONITOR'S ACRONYM(S) <b>AOARD/AFOSR</b>		
			11. SPONSOR/MONITOR'S REPORT NUMBER(S) <b>AOARD-054002</b>		
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release; distribution unlimited</b>					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT <b>Automatic Machine Vision involves humans building machines capable of recognizing objects and scenes in digital images without further human assistance. Machine vision is a bottleneck in robotics and automated systems. When human programmers construct vision systems they are usually designed so that the program architecture and the data are optimized for the particular problem and classification technique being used. In general machine vision systems are hand-crafted to give the best results for a particular application, but are brittle and perform poorly outside their narrow specification, and lack any ability to adapt. On this project we have been researching a method of creating flexible machine vision systems that can modify their behavior and evolve in particular environments to recognize anything that an operator has indicated as being ?interesting? in that environment. For example, Figure 1 shows typical objects that a house-tidying robot might encounter during its everyday duties</b>					
15. SUBJECT TERMS <b>Image Processing, Machine Vision</b>					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES <b>37</b>	19a. NAME OF RESPONSIBLE PERSON
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>			

# **Construct Abstraction for Automatic Information Abstraction from Digital Images**

Jeffrey Johnson<sup>1</sup> and Masanori Sugisaka<sup>2</sup>

<sup>1</sup>Department of Design and Innovation, The Open University, MK7 6AA, UK

<sup>2</sup>Department of Electrical and Electronic Engineering, Oita University, Japan

<sup>2</sup>Adanced Research Institute for Science and Engineering, Waseda University, Japan

29<sup>th</sup> May 2006

## **Contents**

1. Introduction .....	2
2. A Mathematical Theory of Multilevel Systems. ....	4
2.1 Multilevel aggregations .....	4
2.2 Sets versus relational structure .....	7
2.3 Alpha and Beta Aggregations.....	7
2.4 Numerical Traffic on the Multilevel Relational Backcloth .....	9
2.5 Disaggregation, Segmentation and Intermediate Words .....	9
3. Constraints on Data Structures and Operators.....	13
3.1 Parallel versus Sequential processing.....	13
3.2 The architecture of the hierarchical cone .....	14
3.3 Architectural considerations.....	17
4. Low Level Vision processing .....	18
4.1 Retinal Neurons.....	18
4.2 Traffic-based Pixel Matching Techniques for Recognition .....	19
4.3 Single Level Neural Classification .....	20
4.4. Interpreting the data as constructs.....	22
4.5. Multi-Level Pattern Recognition .....	23
4.6. Spatial Relationships.....	24
4.7 Limitations on the low level matching operations .....	24
5. Higher Level Vision processing .....	25
5.1 The Object-Position Representation Problem .....	25
5.2 Biologically inspired approaches to linear features .....	27
5.3 Synthetic Saccades.....	28
5.4 The Locating irregular object experiment.....	29
5.5 Relational Data Structures and Hypernetworks .....	30
5.6 Operators in the Multilevel Hypernetwork Representation .....	31
5.6.1 Geometry and mathematical constructs in the hypernetworks .....	31
5.6.2 Robust low level gradient run primitives .....	32
5.6.3 Hoffman's Rules .....	33
5.7. Bottom-up and Top Down Dynamics .....	36
6. Conclusions.....	36
References .....	37

# 1. Introduction

*Automatic Machine Vision* involves humans building machines capable of recognising objects and scenes in digital images without further human assistance.

Machine vision is a bottleneck in robotics and automated systems. When human programmers construct vision systems they are usually *designed* so that the program architecture and the data are optimised for the particular problem and classification technique being used. In general machine vision systems are hand-crafted to give the best results for a particular application, but are brittle and perform poorly outside their narrow specification, and lack any ability to adapt.

On this project we have been researching a method of creating flexible machine vision systems that can modify their behaviour and evolve in particular environments to recognise anything that an operator has indicated as being ‘interesting’ in that environment. For example, Figure 1 shows typical objects that a house-tidying robot might encounter during its everyday duties.



**Figure 1. It is proposed that objects be contoured prior to them being analysed and learned**

The intention is that non-programmers can train vision systems by ‘pointing’ at an object in a scene, *e.g.* drawing a contour round it, with the intention that the system will adapt and evolve the ability to recognise such objects automatically. The approach is based on new multi-level combinatorial structures supporting an *architecture* that allows vision systems to generalise and adapt to recognise new classes of objects. This multilevel architecture is based on new combinatorial mathematics in the science of complex systems [6].

There are many approaches to machine vision. These include algorithmic knowledge-based programming, neural network systems that learn from examples, and combinations of both. Many practical systems are based on algorithms or procedures making opportunistic use of special features characterising the particular objects and scenes. Although this may give acceptable performance for a given application, there is usually a poor ability to *generalise* to other similar scenes, and no ability to generalise to different environments. A system designed to inspect industrial parts is unlikely to be incorporated in a mobile planetary robot or to be used to read car number plates.

Thus we seek a machine vision architecture that can:

- use point-and-learn training
- work for cluttered scenes
- adapt to changes in objects and scenes
- adapt to any scene or environment

To achieve this we propose a multilevel architecture in which machine vision systems

- evolve appropriate retinal configurations
- evolve connectivities to represent spatial relationships
- abstract their own higher level constructs
- levels are integrated by new relational mathematics

The key feature of the architecture is the ability of the system to abstract its own constructs from data in a multilevel algebraic representation. This allows the system to learn objects that may change through time, and to adapt to learn radically new objects and scenes without the need to change the underlying program. These requirements are very demanding and beyond any existing machine vision systems.

Our approach to this problem is based on the mathematics of *multilevel hypernetworks*, which generalise network theory to multilevel multidimensional space. Hypernetworks naturally give rise to multilevel systems, and provide the essential structural architecture for self-adapting machine vision systems. This mathematics is introduced in Section 2.

Section 3 considers the implications of the mathematical definitions for the architecture of the system we propose to build.

Section 4 reports our investigations into low level image processing techniques, focussing on analogues of neuronal processing. Algorithmic approaches are also discussed later in the report.

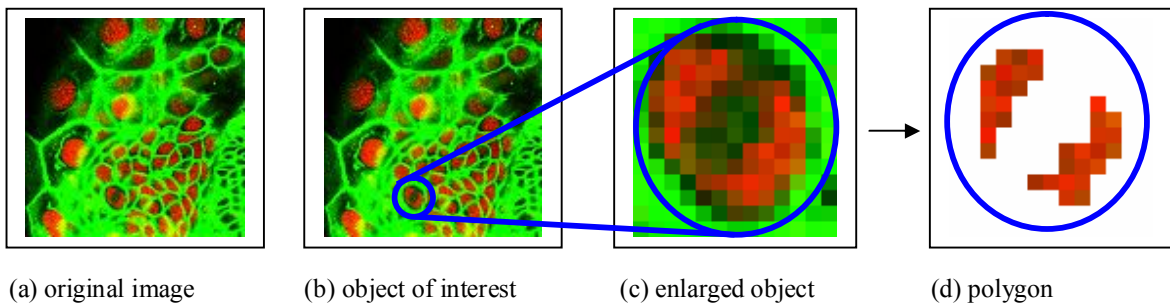
Section 5 discusses higher level vision processing within our architecture, and attempts to present strong evidence that what we propose is feasible.

Section 6 gives our conclusion that, although our objectives are very ambitious, it will be possible to build machine vision systems that can adapt from one field of application another. Also, such systems could be embedded in human-computer systems opening up powerful new areas of application.

## 2. A Mathematical Theory of Multilevel Systems.

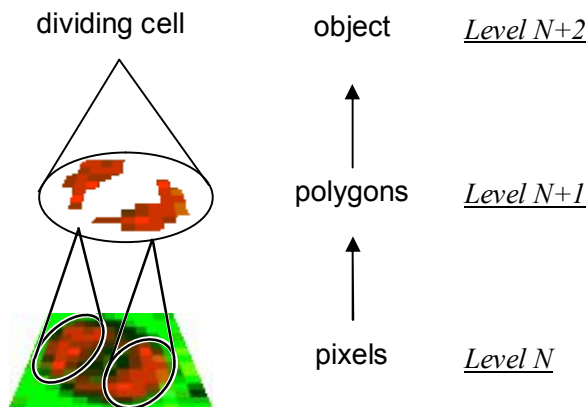
### 2.1 Multilevel aggregations

Except for the simplest images, a single pixel tells us almost nothing. In combination with other pixels it may tell us more. For example, it may be part of a polygon formed from pixels of similar greyscales. The polygon may have a characteristic shape – an *emergent property* not possessed by the individual pixels – that can be identified as an object. Thus pixels aggregate into polygons as illustrated in Figure 2.



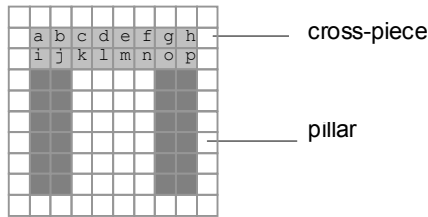
**Figure 2. Similar pixels grouped into polygons.**

The overall scheme is illustrated below, where the pixels exist at an arbitrary lowest level, *Level N*. It is possible to have lower sub-pixel levels but that will not be considered in this report. The *Level N* pixel may aggregate with pixels of ‘similar’ colour to form polygons at *Level N+1*. This aggregation depends on the adjacency properties of the pixels, so the polygons are more than just sets of pixels. They are *structured sets of pixels*. Finally, the two polygons are spatially related to form the ‘dividing cell’ structure at *Level N+2*, which is what the user wants in this case, where the image is one of a sequence of *in-vivo* observations of cell division.



**Figure 3. Machine vision a multilevel systems**

Thus machine vision occurs within a *multilevel system*. We avoid the use of the word ‘hierarchy’ because it has many misleading connotations. In general there are more than three levels. Also, the levels are somewhat arbitrary, and new levels can be defined between existing levels.



**Figure 4. An image of an arch.**

The fundamental idea behind our architecture is that of *n-ary relations*. To illustrate this consider the image of an arch in Figure 4. As we view it, we see two *pillars* supporting a *crosspiece*. The crosspiece, for example, is made up of the pixels marked a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, and p. These sixteen pixels are *assembled* by a 16-ary relation,  $R_{RB}$ , into a rectangular block.

The *construct* ‘rectangular block’ defines a set of objects which will be written  $RB = \{x \mid x \text{ is a rectangular block}\}$ . In order to be operational, this requires a *pattern recogniser*,  $P_{RB}$ , which is able to say of any candidate for membership,  $x$ , that  $x$  is a rectangular block,  $P_{RB}(x) = \text{True}$ , or that  $x$  is not a rectangular block,  $P_{RB}(x) = \text{False}$ .

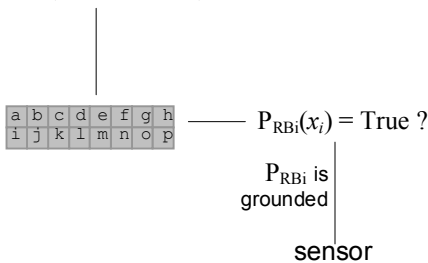
Generally pattern recognisers need to refer to a set of features of the object. In this case there are sixteen features,  $\{x_1, x_2, \dots, x_{16}\}$ , the pixels used to make up the block. Each of these  $x_i$  needs to be of the right type, so the overall pattern recogniser requires a set of sub-pattern recognisers,  $P_{RB,i}$ , with the requirement that  $P_{RB,i}(x_i) = \text{True}$ .

Now it can be seen that the pattern recognition involves two types of decision:

- (i) for each  $x_i$ , it is necessary that  $x_i$  is of the right type, here a dark pixel.  $P_{RB,i}(x_i) = \text{True}$ .
- (ii) given that all the  $x_i$  are dark pixels, it is necessary to established that they are assembled properly so that the relational structure holds with  $P_{RB}(x_1, x_2, \dots, x_{16}) = \text{True}$ .

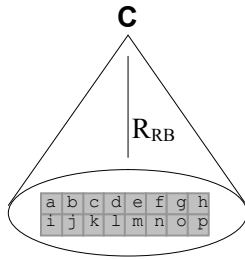
Clearly (i) comes before (ii). There is no point applying expensive pattern recognition procedures to objects which are of the wrong type to form the pattern. However, there is danger of an infinitive regress: To test  $R_{RB}$  it is necessary to test  $R_{RB,i}$  for each  $x_i$ . But to test  $R_{RB,i}$  it is necessary to reduce  $x_i$  to its parts, and test them. And so on. Where can it all end? In robotics and machine vision the answer to this question is easy. Top-down reductionism ends when the pattern recognisers are *grounded* in sensor data. In other words the sensors ‘ground’ everything the machine can know about its environment (Figure 5).

$P_{RB}(x_1, x_2, \dots, x_{16}) = \text{True} ?$



**Figure 5. Reductionist grounding prevents infinite regress in pattern recognition**

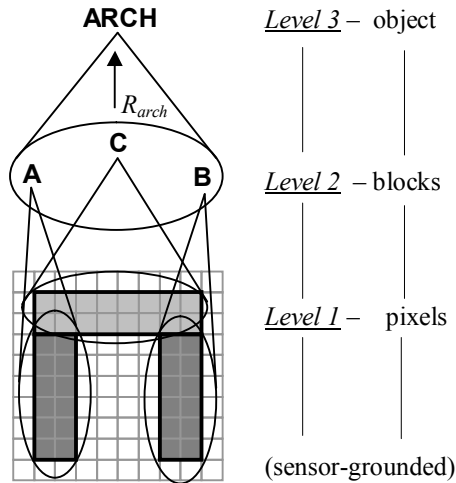
When  $P_{RB}(x_1, x_2, \dots, x_{16}) = \text{True}$  for a particular set of features,  $\{a, b, c, \dots\}$ , we give the resulting object a name, here **C**, and write  $\sigma(\mathbf{C}) = \langle a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p; R_{RB} \rangle$ .  $\sigma(\mathbf{C})$  is called a *simplex* and the elements  $\langle a \rangle, \langle b \rangle, \langle c \rangle$ , etc are called its *vertices*. The parts or features of an object can be said to lie at a *lower level* in its representation than the object itself. If the parts are drawn within an Euler circle (ellipse), the name of the object can be drawn as the apex of a *hierarchical cone*, as illustrated in Figure 6.



**Figure 6. A hierarchical cone.**

The relation  $R_{RB}$  and all its reductionist sub-relations will be called a *construct*. Clearly, in order for a construct to be operational, it must be grounded. Generally constructs are *named*, and they define sets of named objects. E.g., we can use the name ‘rectangular blocks’, and write rectangular blocks =  $\{x \mid x \text{ is a rectangular block}\}$ , which is an *intensional* definition. Alternatively we can write Rectangular Blocks =  $\{RB_1, RB_2, \dots, RB_n\}$ , where each of  $RB_i$  is the name of a particular rectangular block. This is an *extensional* definition.

Figure 7 shows the two stages of assembly of the arch; which is defined as structured set of blocks. The blocks are defined as structured sets of pixels; and the pixels are grounded in reality through the camera sensor.



**Figure 7. Multilevel construct aggregation**

The pillars named as A and B in the image are also structured sets of pixels, as shown in Figure 7. The intermediate constructs A, B and C can be assembled by a 3-ary relation,  $R_{arch}$  to form the construct called an ARCH. Thus we have  $\sigma(\text{ARCH}) = \langle A, B, C; R_{arch} \rangle$ . In this way we build primitive structures from *atomic constructs* (pixels), we build *intermediate constructs* from these

at a higher level in the multilevel representation, and so on, until we recognise objects within scenes at the highest level. At every level we use named constructs to reference the objects abstracted.

This example illustrates a major problem in machine vision. The notions of ‘pillars’ and ‘crosspieces’ are application-specific and social constructs inside programmers’ heads. Vision systems are highly dependent on programmers’ ways of construing the visual universe. It is well known that this can be very different between different people [1], and there is no guarantee that a given programmer will choose the most appropriate constructs. Much better to have the vision system abstract these constructs for itself.

## 2.2 Sets versus relational structure

The cone construction illustrates a number of interesting and important possibilities. Figure 8(a) shows that the same set can be assembled in different ways. Thus the set of vertices alone is not sufficient to represent a simplex. For full knowledge we need to know the relation, which is why we use the notation  $\langle v_0, v_1, \dots, v_n; R \rangle$ , which provides information on both the vertices of the simplex and on the relation that assembles those vertices into the structure.

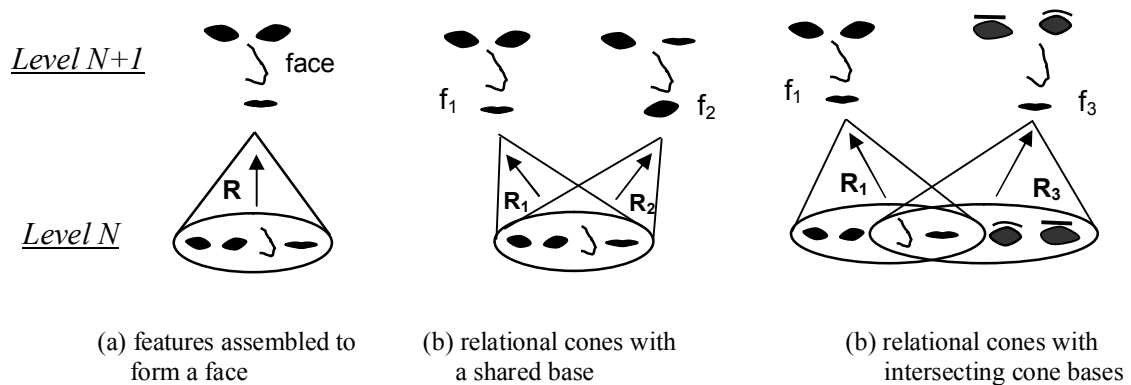
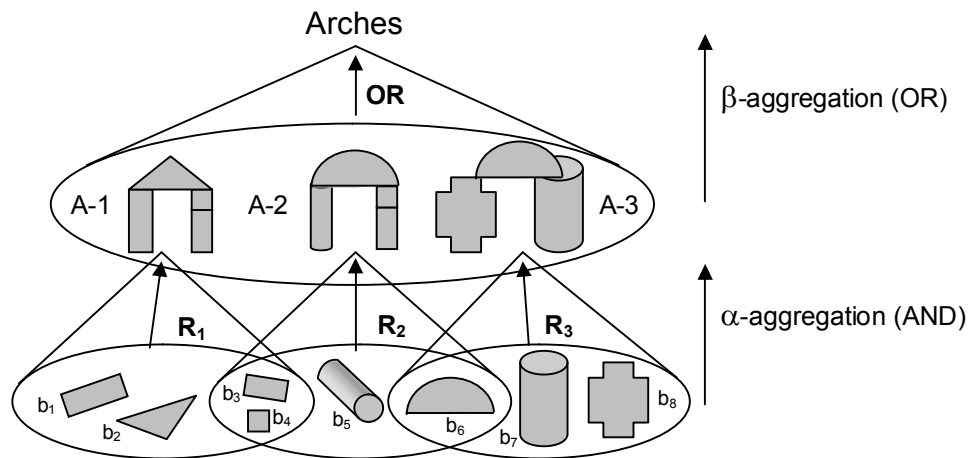


Figure 8. Hierarchical cones representing assembly of parts into holes.

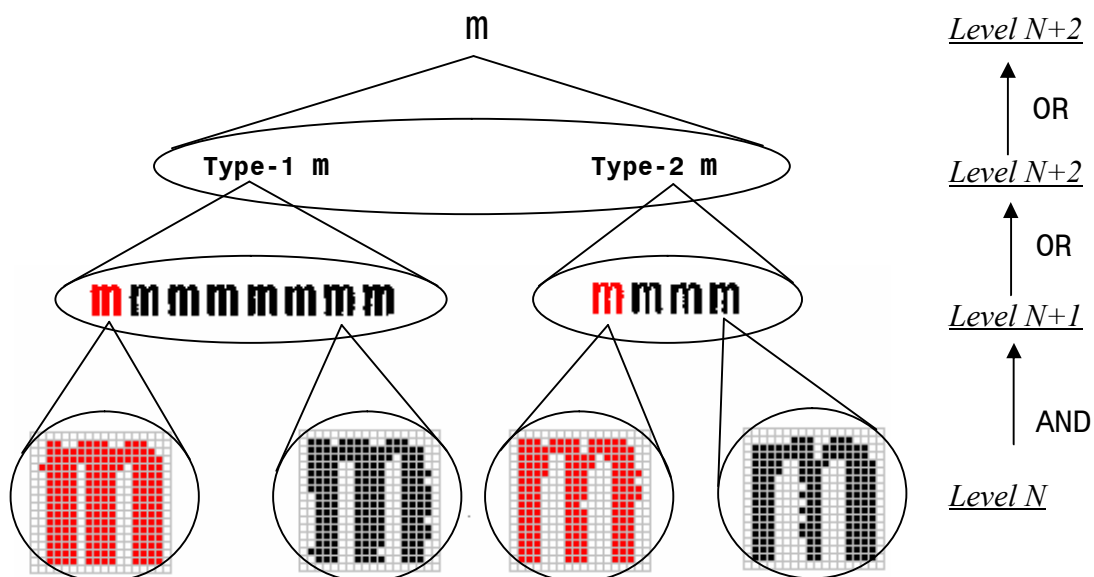
## 2.3 Alpha and Beta Aggregations

The use of  $n$ -ary relations to build objects out of their parts establishes hierarchical levels. However, there is a subtlety in hierarchical aggregation involving another kind of aggregation. This is illustrated in Figure 9 where three arches are assembled from their components. These assemblies require *all* the parts for their  $n$ -ary relation to hold. We call this an  $\alpha$ -aggregation, or an *AND-aggregation*. At the next level the arches are gathered up to form a set. In this case A-1 or A-2 or A-3 is sufficient for an arch. We call this a  $\beta$ -aggregation, or an *OR-aggregation*. Thus the set of arches is defined by a disjunction of conjunctions,  $\vee_j \langle v_{j1}, \dots, v_{jn}; R_j \rangle$ .



**Figure 9. Two different types of multilevel aggregation**

The implications of this for the architecture we are trying to build is that some relations will be assembling new structures between levels (AND) and some will establish *equivalences* on sets of structures. For example, in Figure 10 sets of pixels are aggregated into examples of the character 'm'. The red character on the left was stored in the database, and the other examples next to it were matched and not stored. This set of examples of the 'm' are called 'Type-1 m'. The red character on the right did not match the first m, and was also stored in the database. It determines the set of 'Type-2 m' character. Then either a Type-1 OR Type-2 'm' is sufficient to belong the class of objects called 'm'.



**Figure 10. A double-OR machine vision multilevel structure**

## 2.4 Numerical Traffic on the Multilevel Relational Backcloth

The generality of multilevel systems is that as one moves up the levels, there is a trade off of numbers for structures. In other words, things that exist as individuals at *Level N+k* are counted to become numbers at *Level N+k+1*.

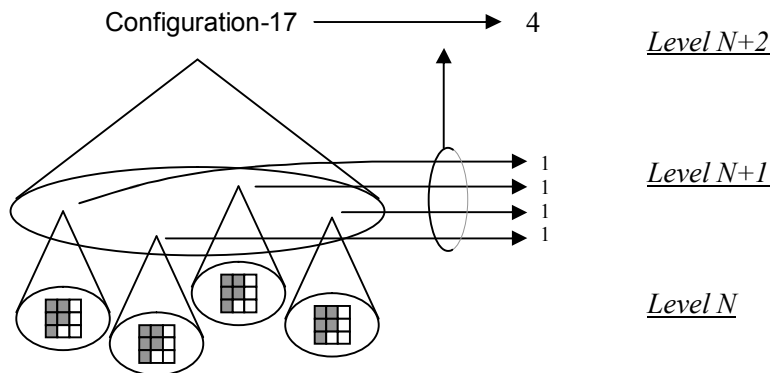


Figure 11. structures at *Level N* represented by numbers at *Level N+2*

In general the multilevel relational structure of a system acts as a kind of relatively *backcloth* supporting the relatively dynamics *traffic* of system activity. In machine vision one is constantly counting things such as numbers of pixels or greyscale distributions. These *patterns* of numbers will be called *traffic* in this report.

Most importantly, in multilevel systems, the traffic at one level must aggregate or disaggregate coherently to other levels. In machine vision and pattern recognition there are both feed-forward and feed-backwards modes where numbers are flowing up and down the multilevel representation.

## 2.5 Disaggregation, Segmentation and Intermediate Words



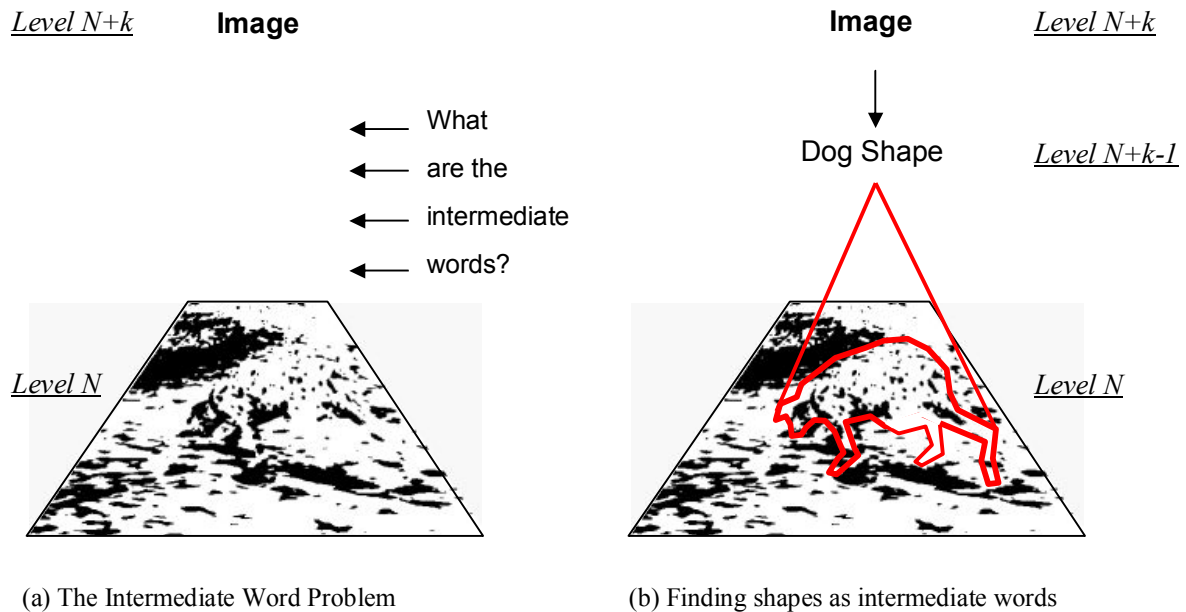
(a) spotted dog illusion



(b) How many pandas?

Figure 12. Extreme problems in segmentation

Machine vision is both a bottom-up and top-down process. Figure 12 shows two images in which there are objects that are difficult to abstract, even for human vision. The problem is to subdivide the image into meaningful parts from which the objects of interest can be reconstructed. For example, in Figure 13(b) a ‘dog shape’ is abstracted as a structured set of pixels.



**Figure 13. Top-down machine vision as the Intermediate Word problem**

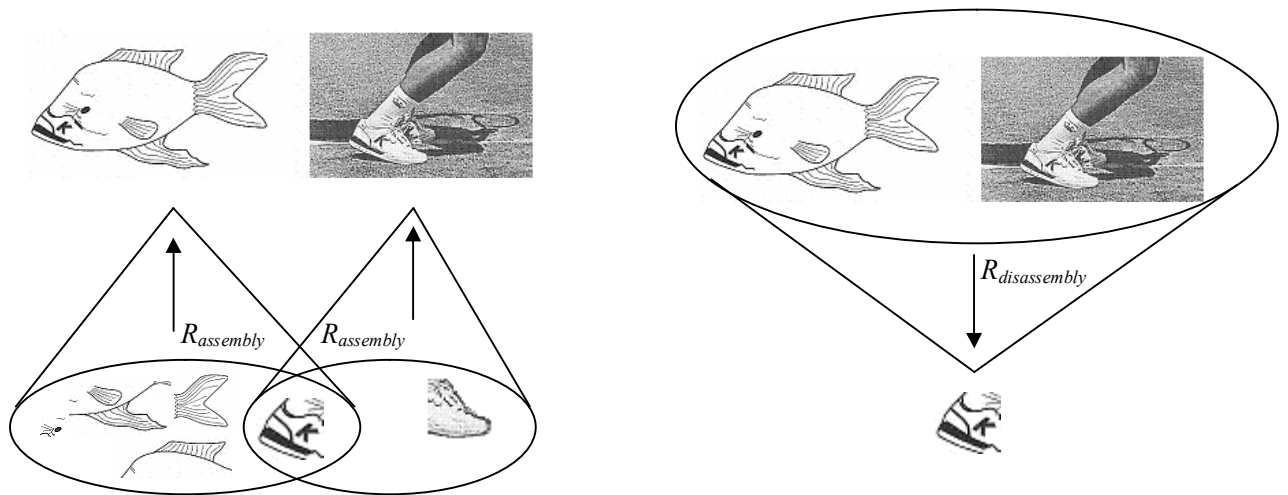
In this example the intermediate words relate to *segments* of the image, and the process of subdividing the image is usually called *segmentation*. Here it illustrates how the multilevel structure is constructed, and why the levels are always relative. Initially the dog shape is abstracted at a lower level to the image at, say, *Level N+k-1*. Depending on the image, there may be other shapes at the level – perhaps another dog or two. The segmentation may progress further, possibly identifying the parts of the dog such as its head, legs, body and tail at a lower level, say, *Level N+k-2*. On the other hand, the parts identified might be aggregated.

For example, if there were other dog shapes in the picture they might be aggregated into a ‘pack’ between *Level N+k* and *Level N+k-1*. This could be called *Level N+k+½* but it is much easier to renumber all the levels. So, the image exists at *Level N+k*, the pack exists at *Level N+k-1* and the dog shape now exists at *Level N+k-2* with parts of the dog existing at *Level N+k-3*. No matter how many levels of description are required, new intermediate levels can always be inserted and the levels renumbered.



**Figure 14. Segmentation and aggregation as complementary top-down and bottom up processes**

In general there is an ordering between the levels defined by the ‘part of’ relation. Then if  $x$  is ‘part of’  $y$ ,  $x$  is at a lower level in the multilevel system to  $y$ , and  $y$  belongs to a set of parts that aggregate into  $x$  which we illustrate with the cone construction seen earlier. In particular the cone construction has an upwards arrow, indicating the aggregation process, and another downwards arrow indicating the segmentation process.

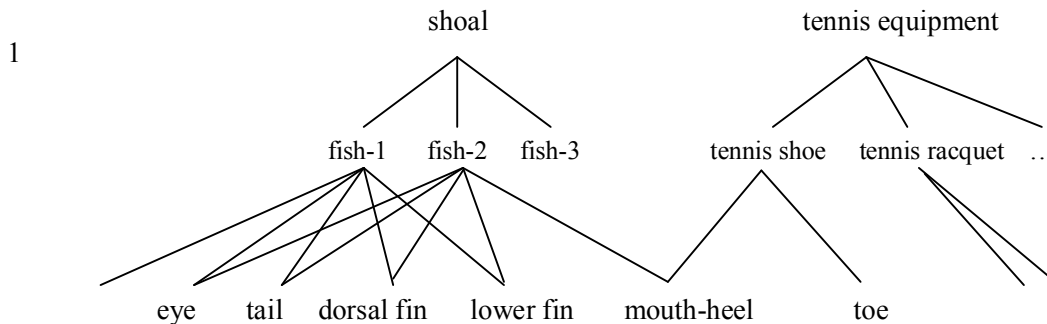


(a) bottom-up aggregation of parts into wholes

(b) top-down disaggregation to common parts

**Figure 15. The part-whole structure is a quasi-order with a lattice hierarchy**

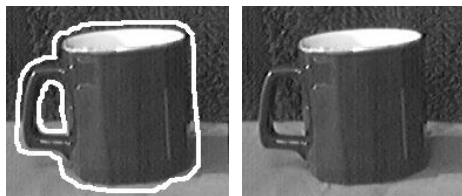
In our theory we make a distinction between objects and features and the *names* of objects of objects and features. For example, in Figure 15 the parts of the fish could be named the ‘mouth’, the ‘dorsal fin’, the ‘lower fin’, the ‘eye’, and the ‘tail’. Similarly, the shoe is made up of the ‘heel’ and the ‘toe’. Allowing things to aggregate into more than one higher level object creates a *lattice hierarchy* which has a richer structure than a conventional tree hierarchy (Figure 16). This multilevel ordering is a *quasi order* (reflexive and transitive but not anti-symmetric ).



**Figure 16. A lattice hierarchy has more structure than a tree hierarchy**

One of the most difficult tasks in machine vision is to *segment* a complex scene into ‘relevant’ parts. Generally one seeks areas that contain discrete objects, such as the coffee mug shown in Figure 17. In Figure 17(a) we show a training object identified by a user. This low-skilled method of teaching the system is the only kind of input the trainer gives. Following this, the system has to find discrete objects in the image to be recognised as of the same type as the training items.

Figure 17(b) illustrates the many problems involved in segmenting images. The mug has no well defined contour, since neither it nor the background are homogeneous in greyscale. In some places the mug merges into the dark background, while in other places it is a relatively light grey due to the reflected light.



**(a) user defined object (b) how to segment?**

**Figure 17. The segmentation problem**

Although the white ellipse is a strong signal to humans that this is a cylindrical object, the machine knows nothing of this *a priori*. In many places the mug has highlights, making it visually very variable. In the first instance we do not assume that that our system will have top-down context knowledge such as ‘if the scene contains an ellipse, then it contains a cylinder’.

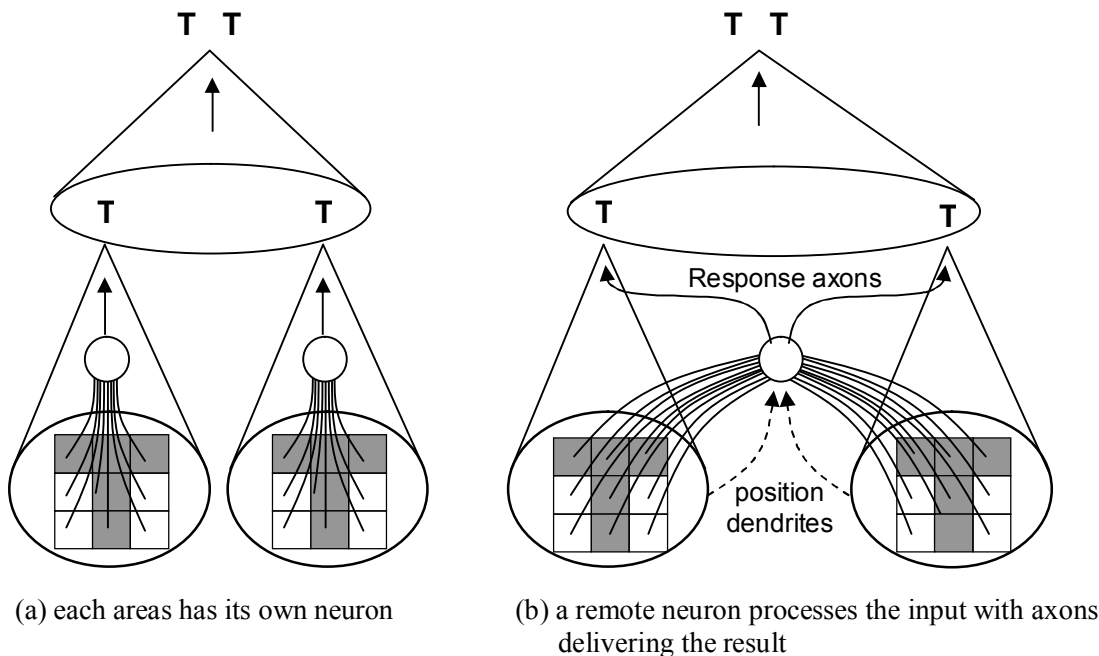
Many of the examples in this paper have been pre-segmented binary images, the methods developed here are highly applicable to greyscale and coloured images. The relational method can be very powerful when, for example, the ‘satellite’ pixels are compared to the centre and classified as light/darker. This approach can lead to areas with varying greyscales but homogeneous greyscale gradient. This has been successfully applied in scientific measurement systems. The details are beyond the scope of this paper, but further details can be found in [7][8].

### 3. Constraints on Data Structures and Operators

The mathematical structure sketched in the previous section has implications for the system we want to implement. In particular all operators implementing hierarchical cones must be *generic*, and must not appeal to system specific properties.

#### 3.1 Parallel versus Sequential processing

Our method assumes that neuronal responses at one level are *assembled* to give a neuronal response at the next, as illustrated in Figure 18. This raises the question whether each local set of inputs has its own neuron to process the data locally, or whether the information is communicated to a remote specialist neuron for processing, with the result brought back by a ‘return axon’ to retain position information. This second case presumably requires a ‘position dendrite’ to provide the position information.



**Figure 18. The local versus remote processing problem**

Option1: Human vision appears allows us to identify objects at very fine resolution, of the order of pixels. Thus for the configuration of Figure 18(a), with spatial information implicit in the hardware, each input pixel is fully equipped with all the pattern recognition machinery necessary to recognise everything. This seems very unlikely.

Option 2: The specialised neuron configuration of Figure 18(b) requires spatial information to be communicated explicitly to the neuron in order to switch on and off the correct response axons. Pursuing a strictly biological analogy in which the firing of the neuron is communicated by all axons, it is necessary to introduce some more hardware to do this switching. However, it seems that in humans the brains undertakes different vision processing tasks in different parts, suggesting that the processing is indeed done remotely.

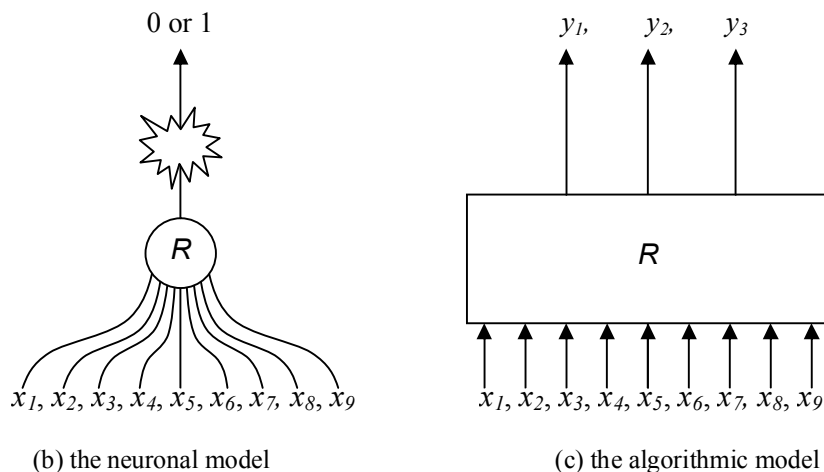
There is the question of how such hardware would process a scene containing many for the same item. Suppose there was a 100 x 100 array of T shapes, with a few missing. Would a single specialised T-recogniser process them all simultaneous? And even if it could, how would it process the position so that the right locations correctly got the 'T-recognised' message?

We call this the *local-remote processing problem*. Our experiments show that Option 1 gives a powerful method of processing images to recognise simple objects, but it fails due to combinatorial explosion. Option 2 has the problem that, within our architecture, the data has to be sent to the processing unit and the result returned, and the mechanisms for this are not obvious.

At this point in the research we had another idea, based on saccades in biological vision. Our idea was that the vision process could be *dynamic* with the focus at any point in time changing according to random movements initiated by coupling the neural response to artificial muscles that move the focus. There are six such muscles in the human eye<sup>1</sup>. This approach transforms pattern recognition in the vision problem into recognising objects and scenes as time sequences of neural responses corresponding to sub-object recognition. The sequences provide the *relational structure* necessary in our architecture.

### 3.2 The architecture of the hierarchical cone

We have identified two major two computational models for implementing assembly relations,  $R$ . One is what we call the *neural computational model* in which a set of inputs, possibly time dependent,  $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9$  enters the processing element and either causes it to fire or not fire with output zero or 1. The other is the *algorithmic model* in which the inputs are used to calculate various output,  $y_1, y_2, y_3, \dots$  a typical implementation of this is the function or subroutine taking input values and returning output values.



**Figure 19. Models of computation for the multilevel cone**

These models have different properties. The 'pure' neuronal model can only output ones and zeros. In implementations such as the multilevel perceptron, weighted links can convert these

<sup>1</sup> *Eye and Brain*, R. Gregory. Oxford University Press (Oxford), Page 45. 1998

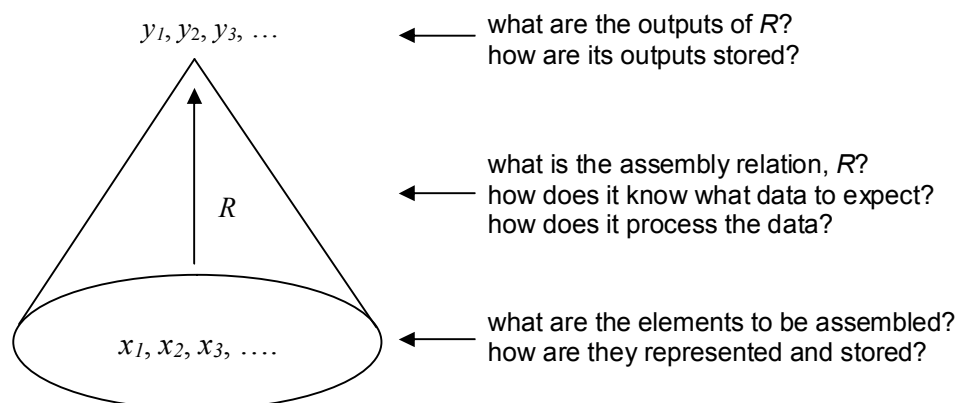
other values, but even there the fundamental assumption is that the neuron either does or does not fire. In contrast the algorithmic model is at first sight more general, since it is able to output any number of ‘independent’ values.

In principle, neuronal implementations, are based on the assumption that the firing conditions can be *learned from examples* which makes this approach very attractive for our purposes. However we have to avoid the naïve assumption that a multi-neuron system such as the multilevel perceptron can accept input from millions of pixels and train to recognise anything. In the first instance such a complicated processing architecture would never converge in practice, and in the second instance there are issues of generalisation and equivalence that cannot be addressed.

Algorithmic models present us with the major problem that as the system moves from one application area to another, even if it were possible, we do not know how to make a system that will create new algorithmic functions and implement them as code. Therefore the most we can hope to achieve using the algorithmic model is to implement *fixed* algorithms that *interpret* data at a high level of abstraction and generality. For example, an algorithm might usefully find a polygon (blob) in an image and calculate numbers such as the statistical properties of the greyscale or dimensions, and output these to feed forward through other processing unit. In contrast to this, we cannot conceive that a system with algorithms *designed* to process low-noise digital camera images could autonomously create new algorithms to deal with, say, high-noise speckled radar images.

The apparent distinction between neuronal and algorithmic processes is less clear than it might appear at first sight. On the one hand, neuronal architectures *are* algorithms when implemented on digital computers. On the other hand, neuronal computers can be designed to produce multiple output weighted outputs. Kolmogorov showed that a system of neuronal units can be configured to take  $m$  general numerical inputs, process its data, and produce  $n$  numerical outputs. Any function  $f: R^m \rightarrow R^n$  can be implemented on such a system. Of course this result is theoretical. In principle *any* function with  $m$  inputs and  $n$  outputs can be implemented on a multilevel perceptron. In practice, *finding* a particular implementation may be impossible because the search never converges.

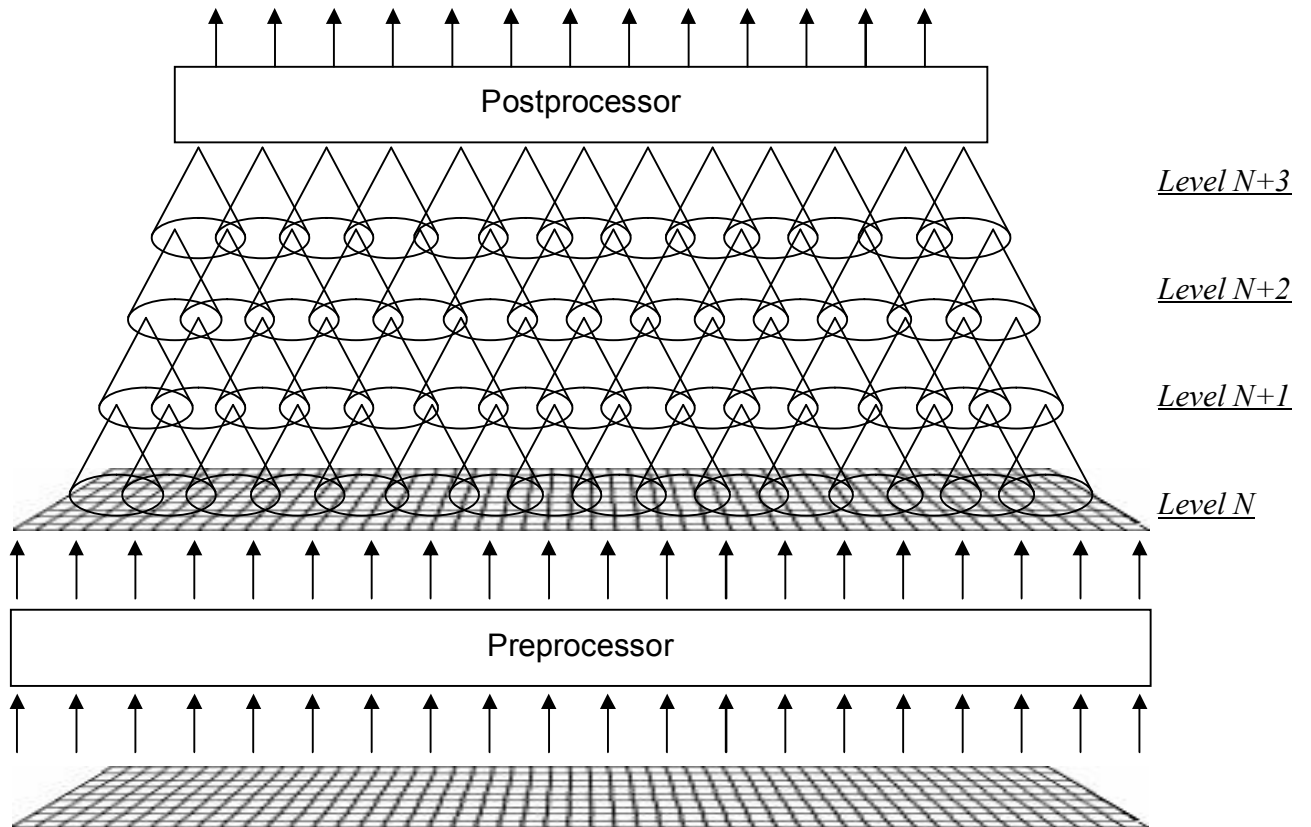
Since our objectives are practical with an *engineering* goal, we make opportunistic use of both neuronal and algorithmic processing approaches. But for either approach there are the questions shown in Figure 20.



**Figure 20.** the hierarchical cone as a general model for hierarchical feed forward processing

We propose that the system be composed of a set of relational aggregators,  $R_i$ , where each  $R_i$  has

- (i) a domain finder,  $D_i$ , deciding what is the set  $x_1, x_2, x_3, \dots$  and related data
- (ii) a computational unit,  $C_i$ , taking data from the domain and processing it
- (iii) a mechanism to place the output of  $C_i$ ,  $y_1, y_2, y_3, \dots$  into the database
- (iv) data structures for the  $x_1, x_2, x_3, \dots$  and the  $y_1, y_2, y_3, \dots$



**Figure 21. The multilevel architecture consists of layers of relational aggregators**

These relational aggregators will be arranged in layers as illustrated in Figure 21. The cones will be both AND and OR aggregations and may be implemented as neuronal or algorithmic processors. Both are constrained by the following rules:

**Computational Rules:**

- (1) The computational model or metamodel underlying a relational processor cannot change. It is hard-wired into the genotype of system.
- (2) The system must *learn*, meaning that its computation at time  $t$  is dependent on computations at time  $t-k$ , and changes to its database.
- (3) There is no high-level symbol system specific knowledge explicitly encoded in the system.

### **3.3 Architectural considerations**

The research described in this paper, in the context of our objectives leads us to the following principles:

Principle 1. Low retinal configurations will aggregate data to form higher level constructs

Principle 2. The constructs will depend on spatial relations

Principle 3. The retinal and higher level configurations should not be constrained by design, but should be allowed to emerge from the images and scenes in its environment

Principle 4. The spatial relations in the system should be implicit in its topology so that Cartesian geometry need not be used (but see Section 5)

Principle 5. Higher level spatial configurations should not be constrained by design, but should be allowed to emerge from the images and scenes in its environment

Principles 1 and 2 are the fundamental theoretical underpinning of our approach. They are supported by algebraic mathematics that can be implemented as data structures in real computers.

Principle 3 is based on the need for the system to adapt to new things. Any system with pre-designed primitives is constrained by what the designer puts in. This *spans* the space of possibilities. Any object not in that space cannot be recognised. This is one reason why conventional machine vision collapses outside its design domain.

If the low level configurations are not to be designed in, where can they come from? We have experimented with forming low level constructs by random configurations of pixels. We have found that generating random masks gives some discrimination between the circular and diamond shapes discussed earlier. However, there remain many open questions, including the optimum diameter for a retinal configuration.

A similar argument suggests there can be no fixed multilevel architecture, and this too must incorporate random processes. Thus the ‘relevant’ configurations of configurations, and the resulting ‘constructs’ have to be discovered by the machine.

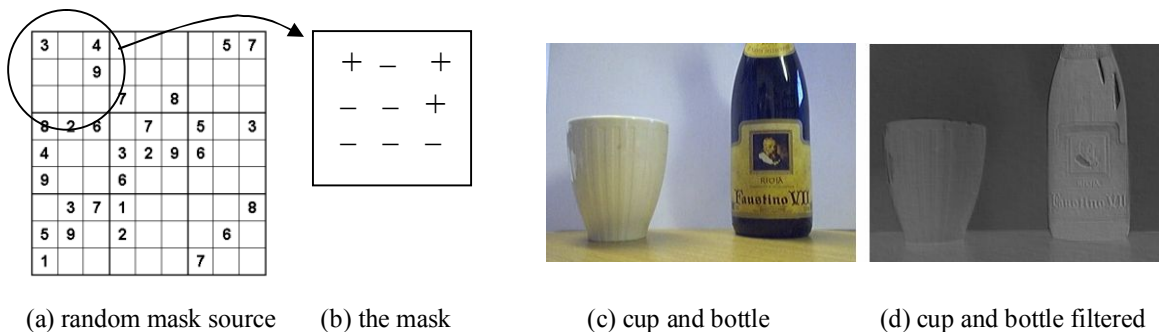
Thus there are two parts to our architecture. The simplest involves the machine learning particular objects and scenes within a given hardware topology. In other words, in the simplest case the machine is fixed, and recognition takes place by values and parameters changing within that structure.

The more demanding part of the architecture involves evolutionary principles to generate and select ‘appropriate’ retinal primitives, and to generate and select appropriate topologies to support relational structure throughout the multilayer aggregation.

## 4. Low Level Vision processing

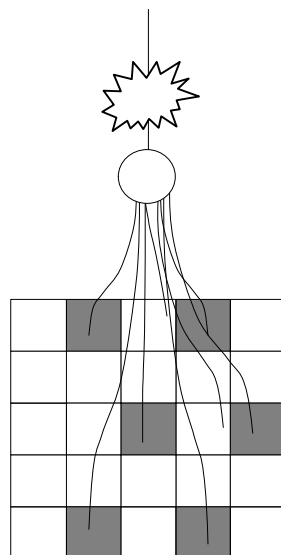
### 4.1 Retinal Neurons

Our approach to low-level machine vision is based on the observation that almost any asymmetric mask will act as an edge detecting filter. For example, Figure 22 shows a Sudoku puzzle taken at random from the Internet. From the top left corner a mask was formed with pluses corresponding to numbers and minuses corresponding to blank squares. This mask passed over the image in Figure 22(c) to give the result in Figure 22(d)



**Figure 22. Random masks filter images**

By a *retinal neuron* we will mean a processing device that takes inputs directly from an image (analogous to the retina in the eye) and processes them. In the simplest case a retinal neuron will 'fire' when it matches a configuration, passing a non-zero value to its output (Figure 23).



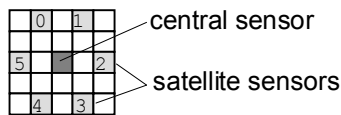
**Figure 23. A retinal neuron**

## 4.2 Traffic-based Pixel Matching Techniques for Recognition

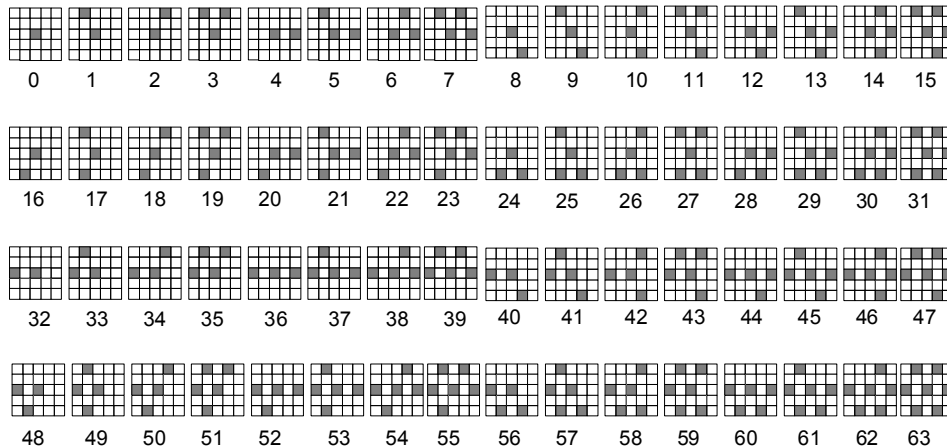
In the proposed multilevel architecture, let the pixels define a base level, *Level 1*. (Lower level sub-pixel constructs are possible (e.g. Johnson and Picton, 1985), but not discussed here. At this level of representation are the usual greyscale histograms.

The next level of representation must be characterised by sets of pixels structured by relations – nothing else is possible! So, *Level 2* in the representation will consist of sets of pixels under  $n$ -ary relations. To illustrate this, consider the pixel configurations shown in Figure 24. To establish them at the lowest level in the representation, these will be called *retinal constructs*.

In Figure 24(a) there is a central sensor, such as light-sensitive rod, responding to relative darkness, surrounded by six other sensors, numbered 0, 1, 2, 3, 4 and 5. There are  $2^6 = 64$  configurations of light-dark for these six *satellite* sensors. The configurations have been designed to have a topology corresponding more closely to packed cells than the usual Cartesian grid. Also they are designed using the ‘next but one’ *neighbours* according to Simon’s three pixel principle [7][8].



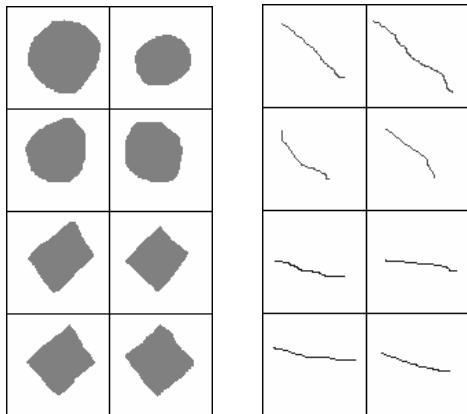
(a) hexagonal array of pixel sensors



(b) the 64 retinal configurations

**Figure 24. Hexagonal pixel constructs**

These configurations are examples of *masks* or *filters* which are widely used in machine vision. As such they have been *designed* by the programmer (me!) and have the problem of subjective selectivity. Although these configurations are attractive for a number of reasons, how can one be sure that they are the most appropriate for any particular objects in any particular environment?



(a) circle and diamonds (b) line segments

**Figure 25. Examples object classes**

The sixty four retinal configurations in Figure 24 were used to analyse eighty hand-drawn shapes, forty ‘circles’ and forty ‘diamonds’, similar to those shown in Figure 25. Each dark pixel in the shapes was analysed by inspecting its surrounding pixels and assigning to it one of the sixty four retinal configurations. As a first level of analysis, the numbers of each configuration were counted, giving a 64-element vector for each configuration. The vectors of the configurations with non-zero frequencies are given in Table 1.

### 4.3 Single Level Neural Classification

Inspection of Table 1 suggests that the frequency vectors alone are sufficient for classification of the simple circle and diamond shapes, and indeed they are. For example, configurations 14 and 31 have much higher

diamond																													
3	4	6	7	12	14	15	24	28	30	31	32	33	35	39	46	47	48	49	51	53	55	56	57	59	60	61	62	63	
2	1	1	22	1	1	21	2	25	24	0	1	2	21	0	1	21	1	2	25	1	21	18	24	2	0	17	24	978	
2	1	1	25	1	2	21	1	14	26	1	0	1	18	1	1	24	1	2	18	0	17	25	27	0	2	23	12	885	
1	0	1	27	2	4	23	0	27	17	3	0	0	25	2	0	25	2	2	28	0	22	28	19	0	4	26	25	1256	
2	0	0	30	1	3	17	2	26	29	0	0	0	22	5	0	28	1	3	28	0	20	21	31	0	3	20	25	1292	
circle																													
3	4	6	7	12	14	15	24	28	30	31	32	33	35	39	46	47	48	49	51	53	55	56	57	59	60	61	62	63	
0	0	2	14	0	8	20	0	25	19	6	0	1	11	31	0	12	0	22	17	0	8	10	14	19	21	6	21	1322	
0	0	0	18	2	18	10	0	6	18	16	0	0	8	33	0	14	2	16	18	0	4	17	13	14	32	15	4	1253	
0	0	2	13	1	11	37	0	10	12	10	0	0	12	28	0	11	1	14	16	0	8	27	15	11	23	24	7	1375	
0	0	1	18	1	14	10	2	14	18	12	0	1	8	23	0	15	1	22	14	0	5	10	12	20	19	9	13	1083	

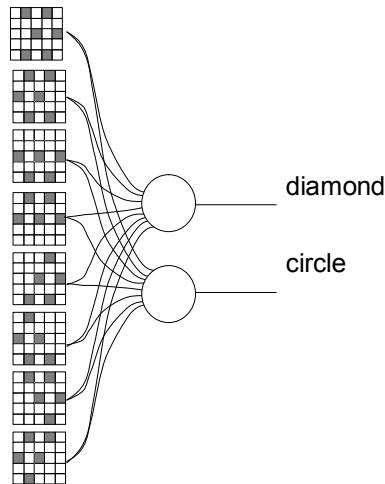
Table 1

**Table 1**

frequency for the circles than the diamonds, reflecting their natural response to vertical left and right edges respectively. Similarly, configurations 7, 30, 51 and 57 favour the diamond shape by responding well to oblique edges.

In principle, a conventional multilayer perceptron neural network will classify such data well, assuming convergence. Note that in Table 1, twenty nine of the sixty four possible configurations

respond to the eighty shapes, leaving thirty five retinal configurations that do not respond to these shapes. Training the network with all sixty four configurations as inputs increases the computation and the possibility of non-convergence.



**Figure 26. A single level vector neural classifier**

Table 2 gives the configuration counts for the line segments shown in Figure 25(b). The response of these objects to the retinal configurations is completely different to that for the shapes. These response vectors can also be used for robust classification between the steep and shallow line segments. It is encouraging that a single layer neural classifier can discriminate these line segments, since it is believed that animal vision uses such primitives.

However, these classifier soon break down when the number of objects to be classified gets large, as is required for recognising a comprehensive set of line segments.

<b>steep lines</b>										<b>shallow lines</b>									
0	1	4	5	8	9	32	36	40		0	1	4	5	8	9	32	36	40	
9	0	13	2	0	0	13	21	2		21	7	5	1	7	4	5	1	1	
3	0	11	1	1	0	12	27	0		32	4	0	0	4	5	0	0	0	
8	1	17	1	0	0	16	15	2		30	10	1	1	10	18	1	0	1	
7	0	11	0	0	0	11	28	0		28	11	1	3	13	10	3	0	1	

**Table 2. Line segments frequencies**

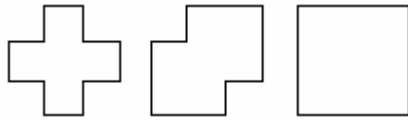
**( Fig 4)**

The approach to pattern recognition illustrated here maps the object to a vector of numbers counting the frequency of ‘interesting’ features of the objects, interprets the vector as a point in multidimensional space, and classifies the points according to some notion of ‘similarity’. In terms of our objectives it begs two questions:

1. where do the ‘interesting’ features come from?
2. is a single level of processing adequate to discriminate objects in complex scenes?

In answer to first question, in our illustrative application, the ‘interesting’ features were designed in by the programmer. Delegating the selection of ‘interesting’ features to a programmer

inevitably means that the system will be limited in its ability to recognise objects, and unable to adapt to recognise objects that are very different from the design specification.




**Figure 27. Shapes with equivalent vertical and horizontal lengths**


It is easy to show that this kind of single level of classification is inadequate in general for object recognition in vision. For example, the objects in Figure 27 all have the same length of vertical and horizontal edges. Conceivably the corners would have different retinal configurations, but the numbers would be small, and robust discrimination between the objects by a single vector of retinal configurations is unlikely.

The answer to the second question must be that a single level of classification is not adequate. If it were, objects and scenes could be presented to a network as an input vector, to deliver recognition of classified objects. Even if this were possible in theory, it would be impractical because combinatorial explosion mean that the necessary input vectors would have astronomic numbers of elements.

#### ***4.4. Interpreting the data as constructs***




In the previous section it has been seen how some retinal configurations can be associated with constructs such as 'oblique', 'vertical', 'left and 'right' edges. These are human constructs that can be imposed on the data. The machine, of course, does not share these constructs explicitly in its representation. Thus there is a co-relation between our concept of a 'round edge' and, say, the pixel configuration 49, , taking a relatively high value for circular objects.

Put like this it becomes possible to understand why conventional approaches to machine vision have failed so comprehensively. As programmers we seek appropriate descriptors or constructs to represent objects to be recognised. We look at an object, and abstract properties such as 'roundness' and 'straightness', that our language conveniently has terms to describe. We then seek machine-based abstractions that match these linguistic constructs.

But as animals we constantly recognise objects for which there is no explicit name. For example, most readers will recognise the shape  as being one of those in Figure 27, even though this shape has no explicit common name. Since I want to talk about it I will give the name of 'double-square shape'. Then I can say things like 'the double square shape is between the cross and the square in Figure 27, and even begin to reason about double-square shapes. However, if such a shape were to be recognised within a machine, it can simply be named implicitly by the data structures, possibly, and its position in memory.

Freeing ourselves from serendipity abstractions in a particular programmer's head, and designing machines to form their own constructs is here seen as the way forward. To some extent this is what multilayer neural networks do, and some researchers assert that each neuron is processing a construct. However, that approach is relatively blunt, since the constructs are always implicit.

## 4.5. Multi-Level Pattern Recognition

Figure 28 shows one of the diamond shapes of Figure 25 with its numbered retinal pixel responses letter-coded as follows: A (configuration 7 in Figure 24), B (14), C (15), D(24), E (25), F (30), G (35), H(47), I (49), J (51), K (53), L (55), M(56), N (57), O (61), P (62), dot (63), and X (all others). Note the sequence ACH (    ) recurs along the bottom left edge. This can be written as a simplex,  $\langle A, C, H; R_{\text{horizontal}} \rangle$ , in the terms of Section 2. This can be considered to be a configuration of retinal configurations, and exists at a higher level of aggregation.

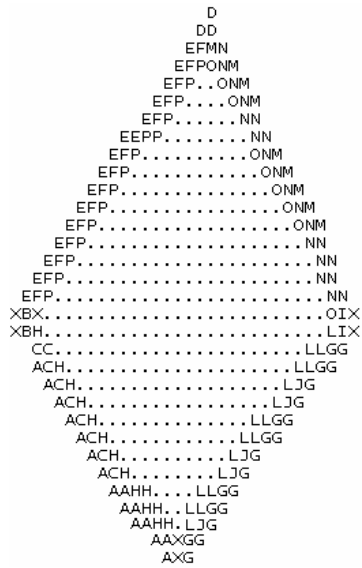


Figure 28.  $\langle A, C, H; R_{\text{horizontal}} \rangle$  as an emergent construct

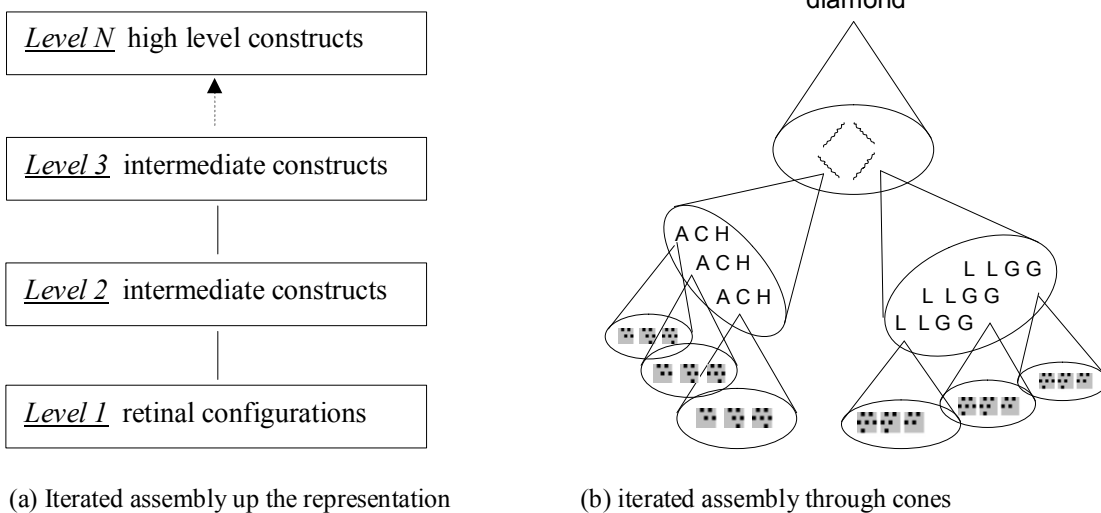


Figure 29. Multi-level aggregation

The ACH configuration emerges from the diamond shape, Let it be denoted  $\sigma_{ABC}$ . Then pairs of such configurations can form the structure  $\langle \sigma_{ACH,i}, \sigma_{ACH,j}; R_{\text{above\_left}} \rangle$ , at yet another level of

aggregation. Let this structure be denoted by the symbol  $\sigma_{ACH}$ . Then these too can be aggregated to form a structure that eventually aggregate into structures involving all seven of the ACH sequences. From a human perspective this could be called a *straight edge*. From a machine perspective this is a learned or evolved structure, physically embedded in the machine that has emerged because there is advantage in it doing so.

## 4.6. Spatial Relationships

The configurations in machine vision inherently involve spatial relationships. Conventional approaches to machine vision often take a highly geometric approach to spatial relationships, based on the Cartesian geometry of the pixel grid. It is interesting to consider whether Cartesian geometry is a product of the human mind, or part of the fundamentals of its workings.

From our perspective it is much easier, in principle, to represent multilevel spatial structure through multilevel tessellations and connections between levels. In other words, we propose to proceed on the basis of spatial relationships being hard-wired into the substrate of the vision system. This idea is illustrated in Figure 30, where four objects have been recognised, and the spatial relationship between them is established (and computed) by located connections between the site of response and higher level processing that recognises the object.

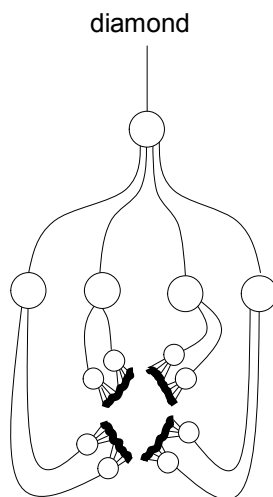


Figure 30. Hard-wired spatial relationships

## 4.7 Limitations on the low level matching operations

Although they give good results in many cases, these low level pixel matching operations have considerable limitations. In particular they are not invariant to any of the following transformations

translation	(images have to be focussed under the mask)
scale	(images can be normalised for scale)
rotation	(not invariant – rotated copies must be used)
sheer	(not invariant)
perspective	(not invariant)
topological	(not invariant)

However, in cases where the object is simple, these low level masks can be very useful.

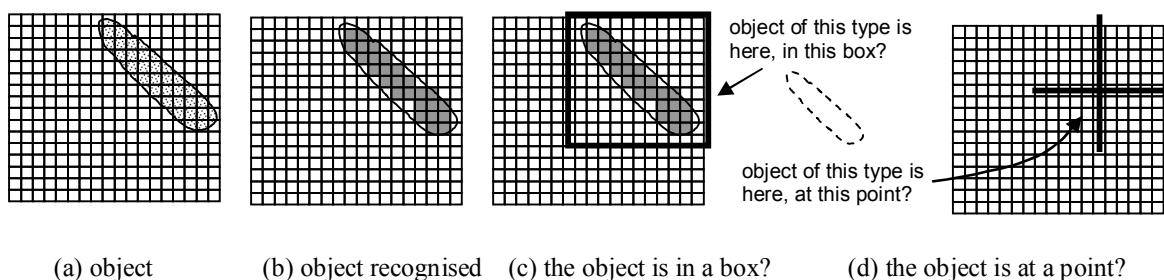
## 5. Higher Level Vision processing

### 5.1 The Object-Position Representation Problem

One of the greatest difficulty in this research has been trying to understand how higher level objects can have their *position* represented within the multilevel representation. The position of lower objects is less problematic due to their *local* nature. However, for large objects in images there is the question of where they are and, more pertinently, how to represent their position.

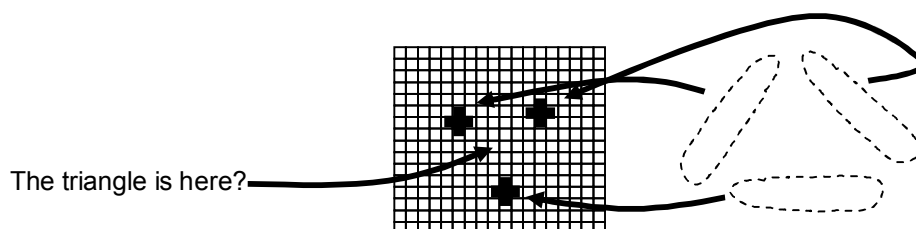
In machine vision various methods are used to locate objects. One common method is draw an enclosing box with the object located, by definition, at the centre of the box. This is the approach underlying the low-level processing in the previous section.

The problem addressed at the higher level is illustrated in Figure 31. Here there is an object in the image which we suppose can be abstracted and classified. The question is where is it? Can it only be located within a box? If it were a large irregular object the box could be very big, leaving the position of the object ambiguous within the box. Or can the object be located at a particular pixel point, as shown in Figure 31(d). The problem with this approach is that a large object will have an influence over many pixels, and not just the pixel 'holding' it.



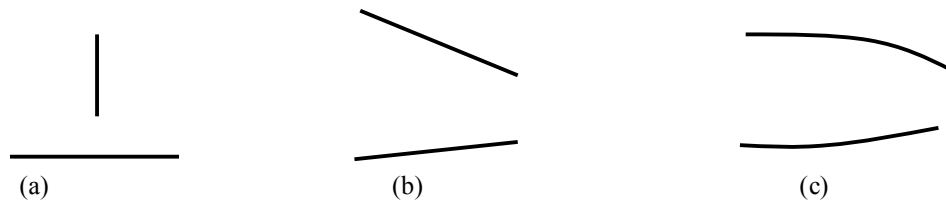
**Figure 31.** The problem that the object can be recognised, but where is it?

Although the solution of representing objects by a pair of Cartesian coordinates is attractive, the problem seems to get worse at higher levels. For example, in Figure 32 three of the line-like objects are recognised at *Level N+1* and form a triangular object at *Level N+2*. Then the three objects are mapped to points and, and the located linear objects can be mapped to a triangular object with position represented at another point?



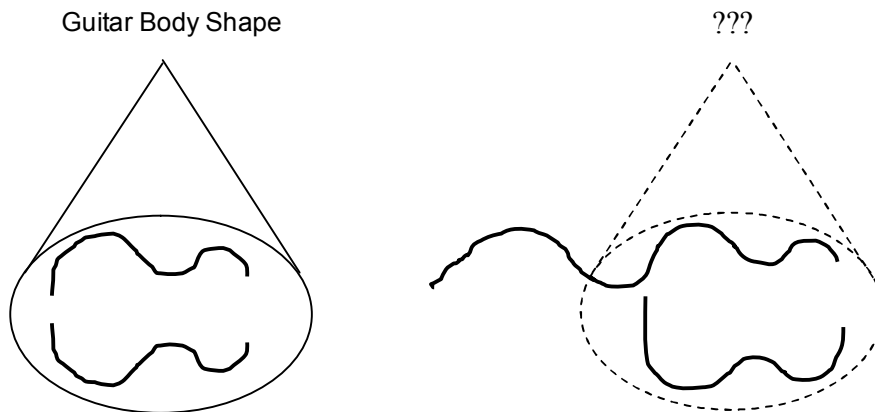
**Figure 32.** Where is the triangle?

Using the Cartesian grid representation, and the method of Cartesian geometry, a straight line is easy to represent as a pair of points, or four numbers:  $(x_1, y_1), (x_2, y_2)$ . This is very convenient for conventional engineering approaches to machine vision, since given two pixels represented by two points  $(x_1, y_1), (x_2, y_2)$  the *line* between them is easy to specify as  $y = (y_1 - y_2)/(x_1 - x_2) * x + y_1 - x_1 * (y_1 - y_2)/(x_1 - x_2)$ , and this can be manipulated in various ways to find points of intersection or to test closeness. More complicated methods allow a wide variety of curves to be represented inside computers, including splines, Bezier curves, and calculus-based methods.



**Figure 33. Biological vision probably does not involve setting up and solving equations**

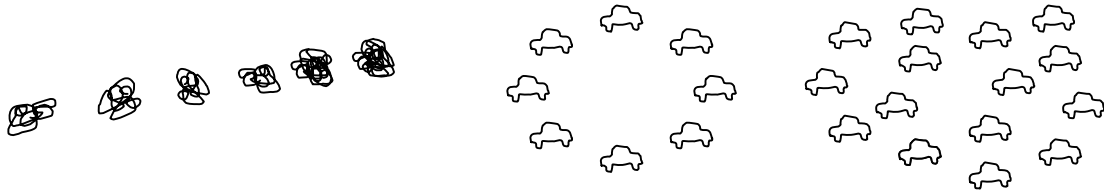
Figure 34 illustrates another potential problem with linear features. In principle curves such as the sides of a guitar are relatively easy to find using conventional methods, and in our architecture, they should be assembled to form higher level structures. When both sides are clearly recognisable, this aggregation may be simple. But it often happens that contours of objects merge with contours in the background, as illustrated in Figure 34(b).



**Figure 34. How can lines be subdivided in the representation?**

If mathematics were at our disposal, important features of images could be calculated such as the intersections of the lines in Figure 34. The problem with this approach is that if used in a self-adapting vision system, the system cannot manipulate the equations by itself. Thus any mathematics like this has to be 'hard-wired' into the system and, apart from changing particular values and parameters, cannot be changed. In contrast to this, biological vision systems probably do not set up mathematical equations, but process the information differently.

## 5.2 Biologically inspired approaches to linear features

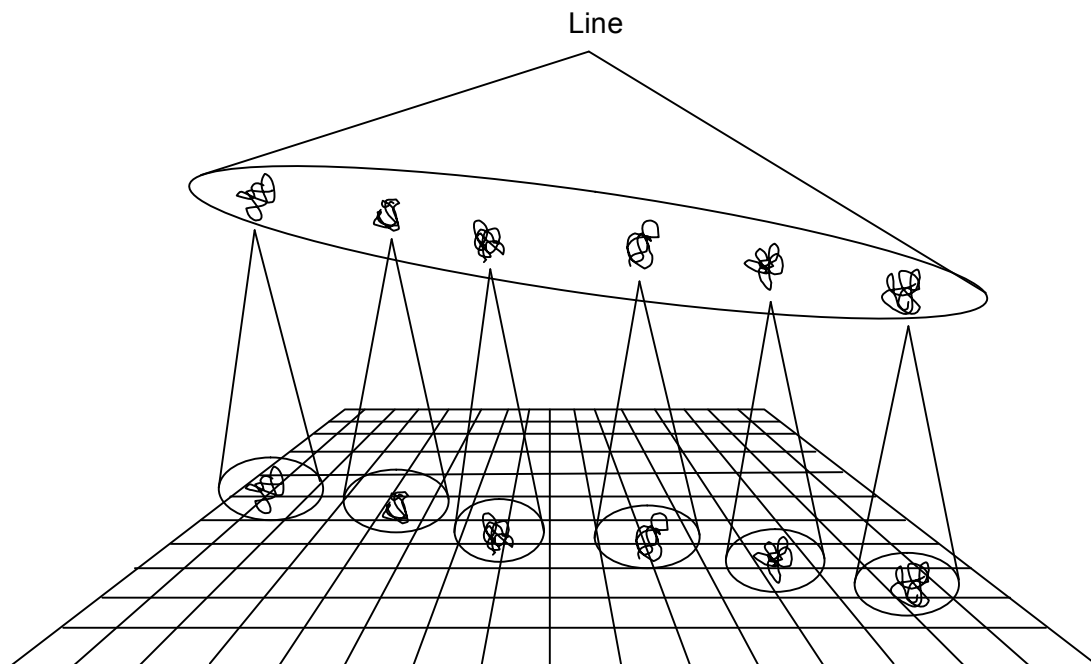


(a) lines can be seen between objects    (b) objects can make contours    (c) more is less for lines

**Figure 35. Human vision constructs lines and contours from configurations of objects**

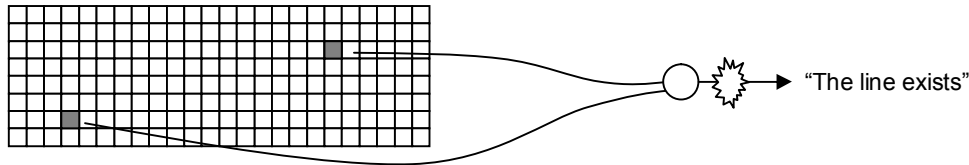
Human vision allows us to see linear features almost everywhere, even when there is hardly any explicit information. In Figure 35(a) a few irregular scribbles clearly forms a linear feature. In Figure 35(b) just eight car shapes form a circle. However, it is not just the presence of features that counts, and sometimes ‘more is less’, as illustrated in Figure 35(c).

These examples make the question of what is and where is a line even more perplexing. Even allowing that the lines and circle can be found in Figure 35, where exactly are they? Even if the co-linear features in Figure 35(a) could be aggregated into a ‘line’ as illustrated in Figure 36, where *is* the line, if it not to be represented in a Cartesian data structure?



**Figure 36. How can lines be represented in non-Cartesian data structures?**

In principle, the problem of representing lines can be solved by using a two-input neuron that fires when the ends of the line are activated, as shown in Figure 37. Of course this is not a practical solution, since it would imply the existence of a line between every pair of activated pixels, and the number of ‘lines’ detected would be astronomic. If two pixels is not enough to signify a line, how many are needed? If all the intermediate pixels were included, the number of masks required to represent possible lines in images would also be astronomic. How then can lines be represented without recourse to coordinate geometry?

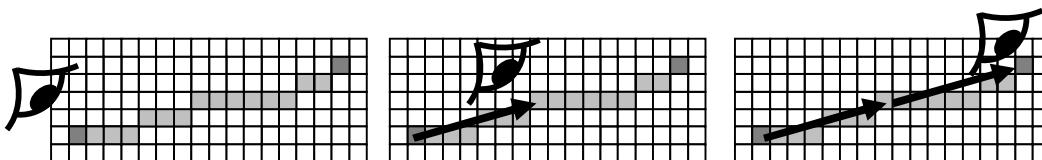


**Figure 37. An impractical ‘solution’ to the line representation problem.**

### 5.3 Synthetic Saccades

This issue of finding a way of representing the geometry of images without using traditional mathematical techniques became a major problem in this research. Even representing structures corresponding to the simple idea of ‘line’ seems very problematic.

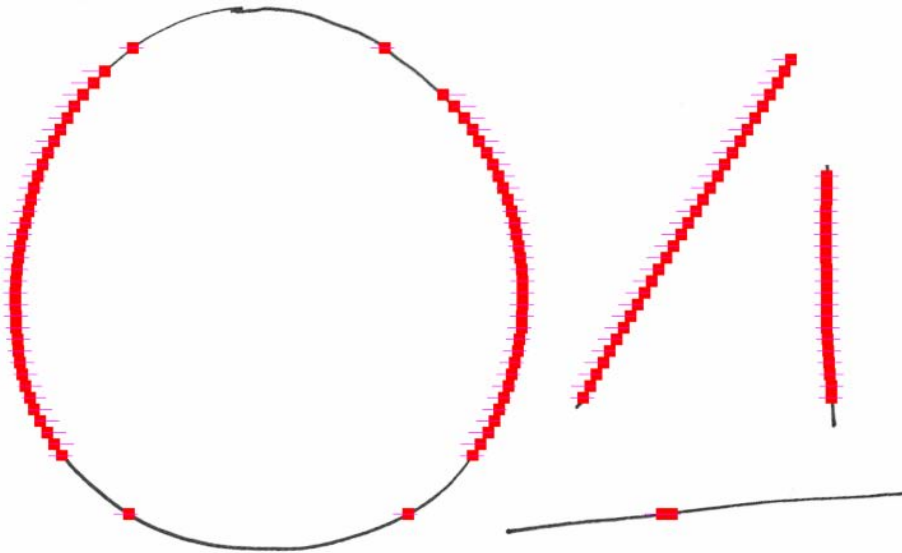
One solution to it appeared late in the project, and has only been partially explored. It is based on the observation that biological vision is characterised by saccades, with the eye moving to focus different parts of the image on the fovea. The sequence of movement of the ‘eye’ then establishes relational structure between the pixels and features.



**Figure 38. Constructing relational structures by moving the focus of vision.**

The human eye has six muscles to move it (Gregory, 1998). We implemented a low-level neuronal approach in which retinal neurons had been trained from a contour image, and a movement impulse was associated with each. The idea was that that the focus of the artificial ‘eye’ should move over the image automatically in response to what the image contains.

This approach moves the representation problem to another level, since the contour then becomes a sequence of neuronal firing which encodes the feature being ‘tracked’.



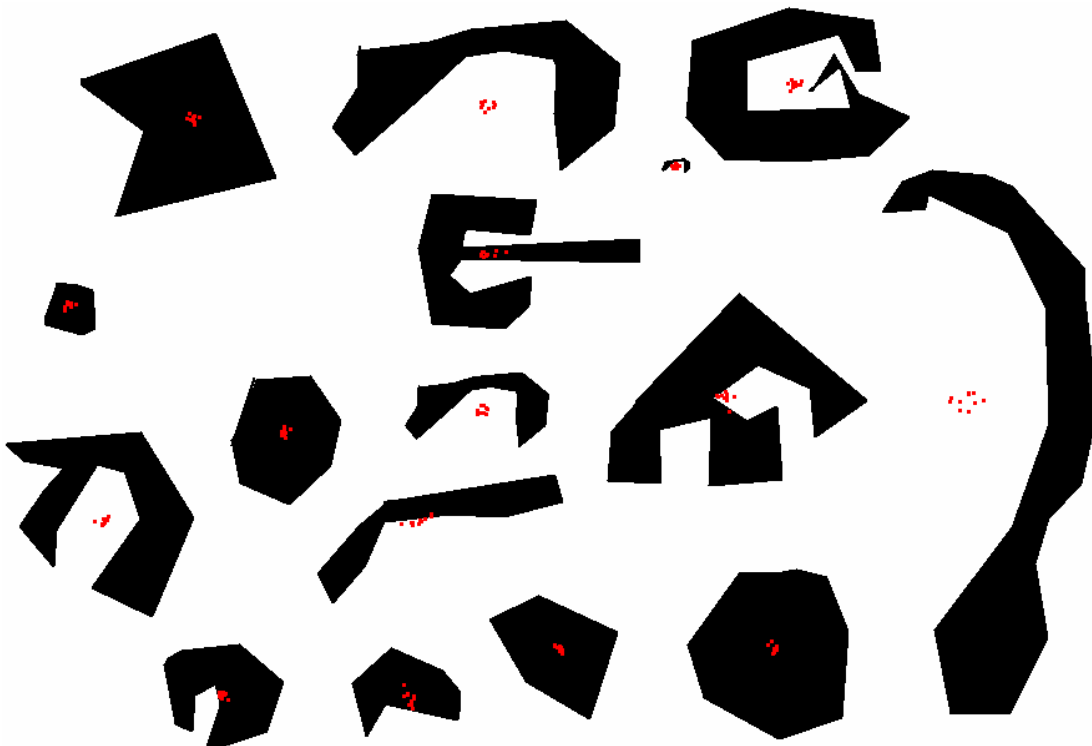
**Figure 39. Synthetic saccades**

Figure 39 shows an implementation of this approach using neurons sensitive to vertical features. As can be seen, vertical features are tracked with great robustness. For clarity in the diagram, neurons sensitive to horizontal features were not used.

Although this approach is potentially interesting from the perspective of biological analogy, and although the tracking was implemented using neuronal detectors linked to virtual muscles, the approach is very similar to contour tracking which has a long and varied history in machine vision. The approach was not pursued further in this phase of the research, but remains a possibility for further investigation.

## **5.4 The Locating irregular object experiment**

In the context of the problem of locating objects, we conducted an experiment to see to what extent the human vision system is able to fix accurately the x-y position of objects. To investigate this we conducted an experiment in which a number of irregular shapes were displayed on a screen. Then the subject (Johnson) placed a mouse at the 'centre' of the objects, moving from one to another so as not to fixate on a given point. The red dots in Figure 40 show the distributions of centres. Rather remarkably, these are very tightly grouped for all the figures, no matter how irregular. This supports the hypothesis that human vision locates objects very accurately, no matter what the shape.



**Figure 40. Locating the positions of irregular objects.**

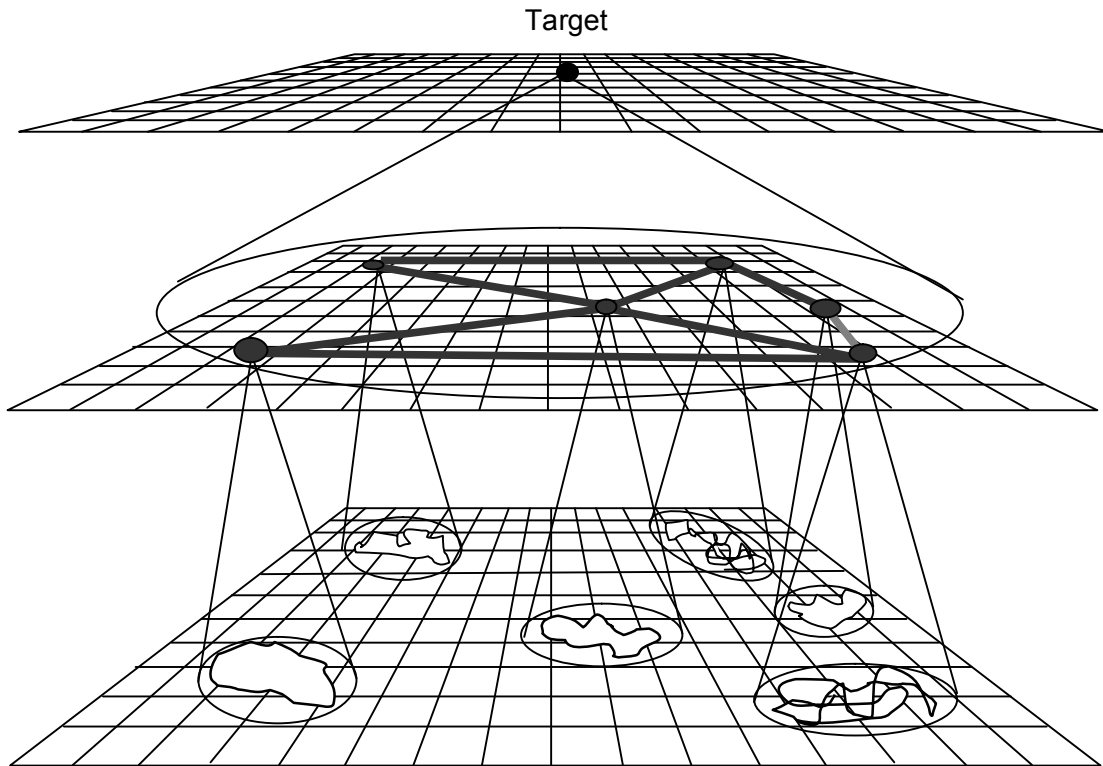
This experiment was pivotal in the research because it suggested that objects can be represented by data attached to precise x-y points, rather than areas as we have discussed previously.

## ***5.5 Relational Data Structures and Hypernetworks***

The strong result of the position-location experiment was not expected and, despite all the disadvantages of the Cartesian representation, suggested that it could be useful and even biologically plausible to represent position by coordinate pairs.

Generally in machine vision, precise geometric matching of objects is rare, since the topological information that everything is ‘connected the right way’ is usually what is required. The hypernetworks introduced earlier in this report are ideal for representing relational structure, and they are fundamental in our multilevel architecture. Whatever the geometry, the higher levels will be characterised by hypernetworks and hypernetworks of hypernetworks. Thus we investigated the use of geometrically registered hypernetworks in which the hyper-vertices are linked to the pixel grid.

The general architecture is illustrated in Figure 41. It is characterised by every vertex at every level being associated with pixel in the original image, thereby linking all objects at all levels of abstraction back to the original image.



**Figure 41. Representing topology and geometry in a multilevel hypernetwork**

## **5.6 Operators in the Multilevel Hypernetwork Representation**

### **5.6.1 Geometry and mathematical constructs in the hypernetworks**

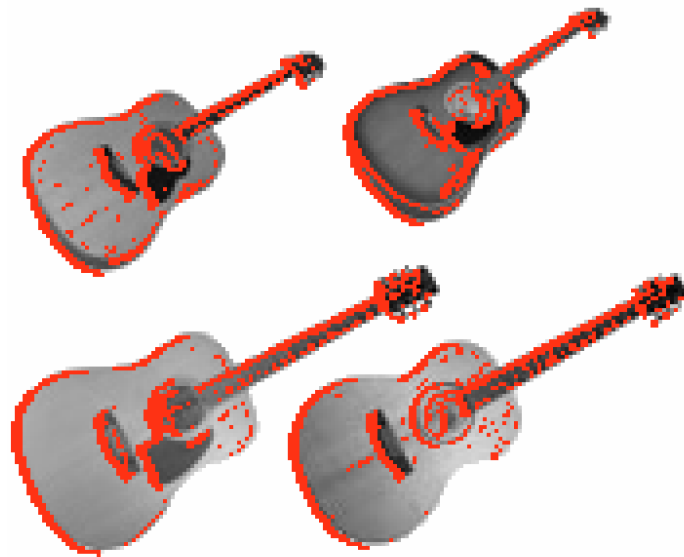
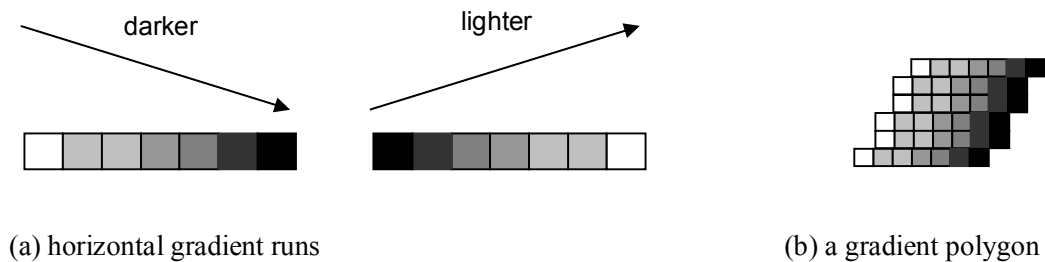
The great advantage of the hypernetwork representation is that it combines geometry and topology. Although the specification of our system precludes application-specific knowledge and opportunistic programming, various operators expressing *general* geometric and topological properties can be hard-wired into the system. Thus for example, it is useful to know when two objects are ‘close’, and to have supporting structure for making such a concept operational in a given application, and allowing the definition to adapt to another application.

We have just begun to investigate such operators, but the combination of geometry and topology offered by geometrical hypernetworks opens up many possibilities.

### 5.6.2 Robust low level gradient run primitives

In our previous machine vision work we have developed many useful concepts, including the notions of *gradient runs* and gradient polygons [Johnson, 1995, Johnson and Simon, 2001]. These are very robust low level features in digital images, and they can be found using both neuronal and algorithmic means.

Figure 42(a) illustrates gradient runs for the horizontal case. They are simply runs of pixels in which the greyscale decrease (left-to-right darker gradient runs) or increase (left-to-right lighter gradient runs). These gradient runs form very robust polygons (Fig 42(b)) in objects where there are features, as illustrated in Figure 42(c).



**Figure 42. Gradient runs and gradient polygons**

Gradient polygons occur in images irrespective of their content, and they are excellent candidates as low level primitives in our architecture. Figure 43 illustrates how gradient runs can be assembled into higher level structures in the hypernetwork.

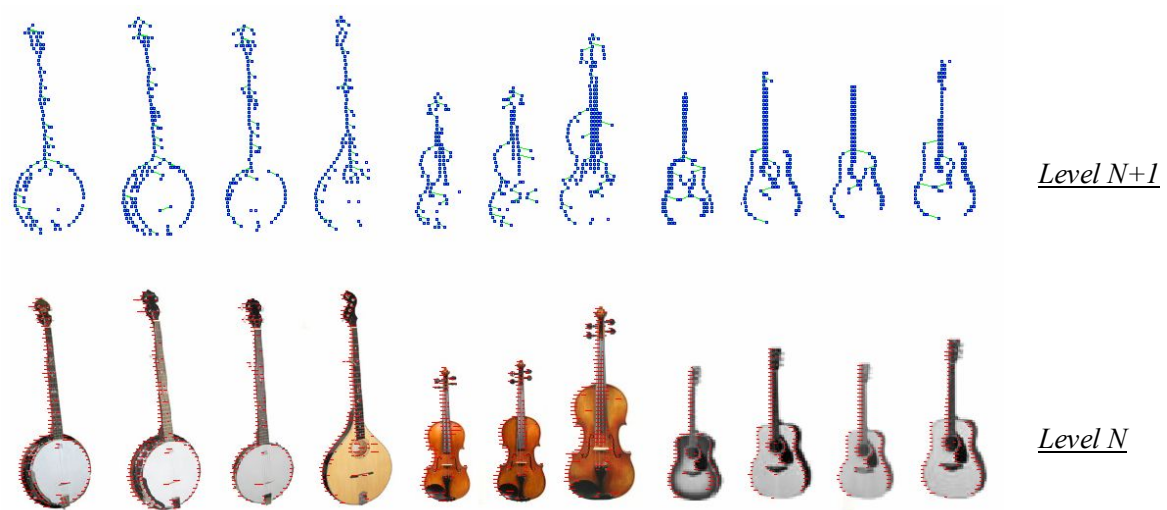


Figure 43. The left-right-darker gradient runs form hypernetwork structures

### 5.6.3 Hoffman's Rules

In his book *Visual Intelligence: how we create what we see* Donald Hoffman [1998] gives a lucid account of how the brain *constructs* what we see from images falling on the retina using a set of interacting *rules*. These rules are behaviourist in that they say what happens rather than giving the details of the computational mechanisms behind what happens. Nonetheless they are very interesting from our perspective because Hoffman's rules apply to *all* images, irrespective of the domain from which they come. Thus this set of rules is entirely consistent with our objective of creating machine vision systems that are not application domain specific. We list these rules below, and illustrate some of the simpler ones. The rules are quoted verbatim, and some are more self-explanatory than others. They are all reproduced for completeness, but to understand them all it may be necessary to consult Hoffman's book.

Hoffman begins with his **Fundamental role of visual rules**: you construct visual worlds from ambiguous images in conformance to visual rules, such as the Rule of **generic views**: Construct only those visual worlds for which the image is a stable (*i.e.*, generic) view, illustrated in Figure 44.

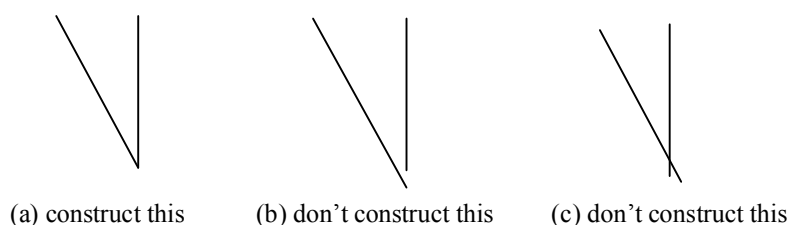
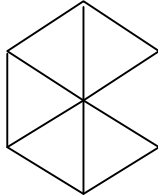


Figure 44.

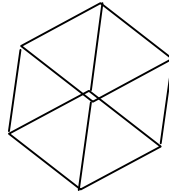
Hoffman's rules are listed below

**Rule 1.** Always interpret a straight line in an image as a straight line in 3-D

**Rule 2.** If the tips of two lines coincide in an image, then always interpret them as coinciding in 3-D



(a) a cube is hard to construct in the Kopfermann figure



(b) a cube is easy to construct

**Figure 45. It is difficult to construct a 3-D vertex in the Kopfermann figure**

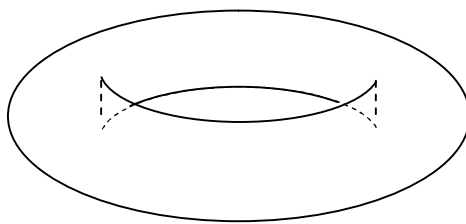
**Rule 3.** Always interpret lines collinear in an image as collinear in 3-D

**Rule 4.** Interpret elements nearby in an image as nearby in 3-D

**Rule 5.** Always interpret a curve that is smooth in an image as smooth in 3-D

**Rule 6.** Where possible, interpret a curve in an image as the rim of a surface in 3-D

**Rule 7.** Where possible, interpret a T-junction in an image as a point where the full rim conceals itself: the cap conceals the stem



**Figure 46. The doughnut image illustrates Rules 6 and 7.**

**Rule 8.** Interpret each convex point on a bound as a convex point on a rim

**Rule 9.** Interpret each concave point on a bound as a saddle point on a rim

**Rule 10** Construct surfaces in 3-D that as smooth as possible

**Rule 11.** Construct subjective figures that occlude only if there are convex cusps

**Rule 12.** If two visual structures have a non-accidental relation, group them and assign them to a common origin

**Rule 13.** If three or more curves intersect at a common point in an image, interpret them as intersecting at a common point in space.

**Rule 14.** *Rule of concave creases:* Divide shapes into parts along concave creases.

**Rule 15.** Minima rule: Divide shapes into parts at negative minima, along lines of curvature, of the principal curvatures

**Rule 16.** *Minimal rule for silhouettes* Divide silhouettes into parts at concave cusps and negative minima of curvature

**Rule 17.** The salience of a cusp boundary increases with increasing sharpness of the angle at the cusp.

**Rule 18.** The salience of a smooth a boundary increases with the magnitude of (normalized) curvature at the boundary.

**Rule 19.** *Salient boundaries:* Choose figure and ground so that figure has the more salient part boundary.

**Rule 20.** *Salient parts:* Choose figure and ground so that figure has the more salient parts.

**Rule 21.** Interpret gradual changes of hue, saturation, and brightness in an image as changes in illumination

**Rule 22.** Interpret abrupt changes of hue, saturation, and brightness in an image as changes in surfaces

**Rule 23.** Construct as few light sources as possible

**Rule 24.** Put light sources overhead

**Rule 25.** Filters don't invert lightness

**Rule 26.** Filters decrease lightness differences

**Rule 27.** Choose the fair pick that's most stable

**Rule 28.** Interpret the highest luminance in the visual field as white, fluorescent, or self-luminous

**Rule 29.** Create the simplest possible motion

**Rule 30.** When making motion, construct as few objects as possible, and conserve them as much as possible

**Rule 31.** Construct motion to be as uniform over space as possible

**Rule 32.** Construct the smoothest velocity field

**Rule 33.** If possible, and if other rules permit, interpret image motions as projections of rigid motions in three dimensions

**Rule 34.** If possible, and if other rules permit, interpret image motions as projections of 3D motions that are rigid and planar.

**Rule 35.** Light sources move slowly

## ***5.7. Bottom-up and Top Down Dynamics***

The multilevel hypernetwork representation that we propose implies a dynamic to pattern recognition in machine vision. In general the object recognition process will be both bottom-up and top-down.

As discussed in Section 2, the relational aggregator has two parts: the set-finder that identifies the potential parts of interesting objects, and the relational tester that ensures the objects are related as they should be.

In the light of the discussion in this section it is possible to see that the conceptually difficult relational tester may be implemented in terms of mathematical operations manipulating the geometry of graphical objects such as gradient polygons.

The set-finders at the lowest level will include algorithmic processors such as those that find gradient polygons. These polygons have many patterns of numbers associated with them, such as their greyscale distribution, and also their two-dimensional grey-scale-gradient distributions [Johnson and Simon, 2001]. These numbers may be used in bottom-up and top-down processes.

Hoffman's rule give a number of ways parts of images can be assembled and disassembled in both bottom-up and top-down processes.

## **6. Conclusions**

In this research we have investigated the architecture of machine vision systems that can adapt from one field of application to another. By hypothesis, the system has no prior system-specific knowledge. It must gain knowledge of specific systems by abstracting information from examples of objects pointed at by human operators requiring very low levels of skill and no computing programming knowledge.

We have identified neuronal and algorithmic processing as appropriate to our system and both types can be integrated coherently. The system is initially bottom-up from pixels to features. It will involve intermediate and high-level operators analysing the geometric and topological properties of the emergent features, and they will play a part in a dynamic bottom-up top-down process.

The fundamental architecture proposed is based on hypernetworks, where these are inherently relational but can integrate objects with geometrical properties. The hypernetwork representation allows relational and numerical data, and supports geometrical operations implemented at high level of abstraction.

The possibility of operating on images at high levels of abstraction is supported by the work of Hoffman, who gives thirty five rules for processing images that allow objects to be constructed in ways similar to the human brain. It is important to stress that Hoffman's rules are application independent.

The research has been undertaken in the context of extensive experimentation. Parts of the overall architecture have been implemented to the extent that they can illustrate the ideas discussed.

From the research we conclude that, although it is a very ambitious objective, that it will be possible to build machine vision systems that can adapt from one field of application another.

Our approach has involved researching systems that are totally autonomous in adapting to new imaging applications. Such systems could be embedded in human-computer systems opening up powerful new areas of application.

## References

- [1] Gregory, R. L., *Eye and brain: the psychology of seeing*, Oxford University Press, 1998.
- [2] Hoffman, D. D., *Visual Intelligence: How we create what we see*, W. W. Norton (New York), 2000.
- [3] Johnson, J.H., 'Some structures and notation of Q-analysis', *Environment and Planning B*, **8**, 73-86, 1981.
- [4] Johnson, J.H., 'Pixel parts and picture whole', in *From Pixels to Features*, J-C Simon (ed), Elsevier Science Publishers (North Holland), 1989.
- [5] Johnson, J.H., Rose, V., 'Autonomous evolutionary machine vision systems', *Proc. AROB 10*, 2005.
- [6] Johnson, J. H., 'Perception', *Mechatronics: designing intelligent machines*, G. Rzevski (ed), Chapter 4, Volume 1, Butterworth (London), 1995.
- [6] Johnson, J. H., 'Hypernetworks for reconstructing the dynamics of multilevel systems', submitted to the European Conference on Complex Systems, Oxford, September 2006.
- [7] Johnson, J.H., Picton, P.D., *Mechatronics: Concepts of Artificial Intelligence*, Butterworths (London), 1995
- [8] Johnson, J.H., Simon, J-C, 'Fundamental Structures for the Design of Machine Vision Systems', *Mathematical Geology*, Vol. **33**, No.3, 2001.
- [9] Marr, D., *Vision*, W Freeman and Company, (New York), 1982.
- [10] Rose, V., Results of experiments in shape-recognition through random selection of sets of pairs of pixels within 75x75 pixel binary images of various simple shapes. Mimeo Open University, 2004.
- [11] Rose, V., Johnson, J. H., 'Shape-recognition using randomly selected pixel pair neuron', *Proc. AROB 10*, M. Sugisaka (ed), 2005.